WHAT IS CLAIMED IS:

1. An implementation of a key-value dictionary shared object for which all concurrent executions of insert-type and delete-type operations thereon are linearizable and lock-free.

2. The implementation of claim 2,
organized as a skip-list-type data structure with associated functional encodings of insert-type and delete-type operations that employ linearizable synchronization operations.

3. The implementation of claim 2, wherein the skip-list-type data structure includes:
plural nodes; and
at least two referencing chains including a first-level referencing chain that traverses the nodes in accordance with an ordering thereof and an Nth-level referencing chain that traverses at least a subset of the nodes in accordance with the ordering.

4. The implementation of claim 2,
wherein a dead pointer indication is employed to mark a node in process of being excised from the skip-list-type data structure.

5. The implementation of claim 4, wherein the dead pointer indication includes one of:
a self pointer;
a pointer to a dead node; and
a back pointer.

6. The implementation of claim 2,
wherein, for the insert-type operations, the employed linearizable synchronization operations include a single-target synchronization primitive employed to introduce a new node into a first-level referencing chain of the skip-list-type data structure.

7. The implementation of claim 2,
wherein, for the insert-type operations, the employed linearizable synchronization operations include a double-target synchronization primitive employed to update an existing node with a new value.

8. The implementation of claim 2,
wherein, for the insert-type operations, the employed linearizable synchronization operations include a double-target synchronization primitive employed to introduce a new node into an N^{th}-level referencing chain of the skip-list-type data structure, wherein N^{th}-level is greater than first-level.

9. The implementation of claim 2,
wherein, for the delete-type operations, the employed linearizable synchronization operations include a double-target synchronization primitive employed to excise a node from a referencing chain of the skip-list-type data structure.

10. The implementation of claim 2, further comprising:
a definition of a node instantiable in shared memory to represent a key-value pair.

11. The implementation of claim 1,
wherein values in the key-value dictionary shared object include literal encodings of values.

12. The implementation of claim 1,
wherein values in the key-value dictionary shared object include references to a literal value or data structure.

13. The implementation of claim 1,
wherein the concurrent executions are mediated by compare-and-swap (CAS) and double-compare-and-swap operations.

14. A method of operating on a shared data structure that includes a representation of an ordered set of keys, the method comprising:

instantiating nodes of the shared data structure in memory, wherein plural levels of same-direction referencing chains traverse respective subsets of the nodes in accordance with a key ordering relationship thereof, a first-level of the referencing chains traversing each node of the shared data structure and at least one other level of the referencing chains traversing less than all nodes of the shared data structure; and

operating on the shared data structure using insert-type and delete-type operations that are linearizable and lock-free for all concurrent executions thereof.

15. The method of claim 14,

wherein the insert-type operation performs a synchronized update of pointers beginning at the first level thereof and continuing upward; and

wherein the delete-type operation performs a synchronized update of pointers beginning at a K^{th} level thereof and continuing downward to the first level.

16. The method of claim 14,

wherein the insert-type operation performs a synchronized update of pointers in accordance with a first succession of the levels; and

wherein the delete-type operation performs a synchronized update of pointers in accordance with a second succession of the levels, the second succession opposing the first succession.

17. The method of claim 14, further comprising:

for a given one of the nodes instantiated, dynamically selecting a number of the plural, same-direction referencing chains that traverse the given node.

18. The method of claim 14,

wherein the shared data structure implements a dictionary.

19. The method of claim 14, wherein values are associated with respective ones of the keys, the method further comprising:

as part of an execution of the insert-type operation introducing a value into the shared data structure; and

as part of an execution of the delete-type operation removing a value from the shared data structure the removed value corresponding to a search key.

20. The method of claim 19,

wherein the correspondence includes a greater-than-or-equal-to key match criterion.

21. A computer readable encoding of a shared object, the encoding comprising:

plural nodes;

plural levels of same-direction referencing chains that traverse respective subsets of the nodes in accordance with a key ordering relationship thereof, a first of the referencing chains traversing each node of the shared data structure and a second of the referencing chains traversing less than all nodes of the shared data structure; and

a functional encoding of linearizable operations on the shared object, wherein the linearizable operations include both insert-type and remove-type operations and are lock-free for all concurrent executions thereof.

22. The computer readable encoding of claim 21,

wherein, on at least some executions, the insert-type introduces an additional node into at least one of the referencing chains.

23. The computer readable encoding of claim 21,

wherein, on at least some executions, the delete-type operation excises a particular node from all referencing chains that traverse the particular node.

24. The computer readable encoding of claim 21,

wherein the delete-type operation employs a greater-than-or-equal-to key match criterion.

25. The computer readable encoding of claim 21,
wherein the shared object implements a shared skip-list-type data structure.

26. The computer readable encoding of claim 21,
wherein the shared object implements a dictionary data structure.

27. In a computational system that employs a shared list-type data structure that includes plural nodes and plural levels of referencing chains that traverse respective ones of the nodes in accordance with an ordering thereof, wherein a higher-level one of the referencing chains traverses no more than a subset of the nodes traversed by a lower-level one of the referencing chains, a method of facilitating lock-free concurrent operations on the shared list-type data structure, the method comprising:

deleting a node from the shared list-type data structure by excising the node from successive ones of the referencing chains, beginning with a highest-level one of the referencing chains that traverses the node and continuing through a lowest-level one of the referencing chains, wherein each such excision employs a linearizable synchronization operation to bridge the excised node and associate a dead pointer indication therewith; and

inserting a node into the shared list-type data structure by introducing the inserted node into one or more of the referencing chains, beginning with the lowest-level referencing chains and continuing through successive zero or more higher-level referencing chains.

28. The method of claim 27,
wherein all concurrent executions of the deleting and the inserting are linearizable and lock-free.

29. The method of claim 27, wherein the dead pointer indication includes one of:

a self pointer;
a pointer to a dead node; and
a back pointer.

30. A method comprising:
defining a shared list-type data structure that includes plural nodes and plural levels of same direction referencing chains that traverse at least some of the nodes in accordance with an ordering thereof, wherein a higher-level one of the referencing chains traverses no more than a subset of the nodes traversed by a lower-level one of the referencing chains; and
inserting into, and deleting from, the shared list-type data structure, wherein all concurrent executions of the deleting and the inserting are linearizable and lock-free.

31. An apparatus comprising:
a definition of a skip list instantiable in storage; and
lock-free means for coordinating concurrent and linearizable executions of both insert-type and delete-type operations on the skip list.

32. The apparatus of claim 31, further comprising:
the storage.

33. The apparatus of claim 31, further comprising:
plural processors, the insert-type and delete-type operations executable thereon.